Web Technologies for Scientific Hearing Experiments and Teaching - An Overview

Bastian Bechtold
Jade Hochschule Wilhelmshaven/Oldenburg/Elsfleth
bastian.bechtold@jade-hs.de

Stephanus Volke
Jade Hochschule Wilhelmshaven/Oldenburg/Elsfleth
stephanus.volke@jade-hs.de

Joerg Bitzer
Jade Hochschule Wilhelmshaven/Oldenburg/Elsfleth
joerg.bitzer@jade-hs.de

ABSTRACT

Scientists of many audio-related fields need to verify their theories by conducting controlled experiments with human test subjects. The process of developing and conducting such experiments often poses non-trivial challenges to scientists and test subjects. Web technologies promise simple delivery of experiments as interactive websites, possibly even on subjects' own computers. Similar benefits are possible for teaching science. While many tasks in hearing experiments and teaching are well-supported with current web-based tools, support for scientific data structures, signal processing operations and statistical data analysis methods is still incomplete in comparison with entrenched non-web tools. These shortcomings could easily be overcome with a few libraries, and would provide a great boon to scientists and educators.

CCS Concepts

• **Applied computing** → **Physical sciences and engineering; Engineering;** *Computer-assisted instruction; Interactive learning environments;* • **Hardware** → Digital signal processing;

1. INTRODUCTION

Scientists need to verify their theories by conducting controlled experiments. Audio-related sciences, such as signal processing, audiology, or hearing technology often rely on human test subjects for these experiments. This affords many non-scientific challenges such as how to deploy research software effectively to human subjects. Existing solutions [5, 8, 10] are often proprietary, complicated, or need direct supervision by the experimenters.

Web technologies can ease the deployment and implementation of hearing experiments significantly. Every computer equipped with a modern web browser and a suitable set of headphones can be used for hearing experiments, and experiments can even be conducted on uncontrolled computers in the subjects' homes. This obviates the need for expensive and hard to install scientific software packages such as MATLAB [16], R [11], SPSS [3], or Scientific Python [7, 19] for hearing experiments, and simultaneously ensures a standardized test environment in the web browser's sandbox.

Furthermore, modern web-based graphing frameworks [1, 21] and data analysis toolkits [2, 18] allow professional visualizations of test results, rivaling those of the aforementioned scientific software packages. Integrated software packages such as these are currently being developed by multiple research institutions [13, 24], with impressive results.

Another crucial application of web technologies is science teaching, which currently predominantly relies on static lectures and programming exercises. Current teaching aids for signal processing and hearing research often involve example programs and programming challenges. Web technologies, such as web apps for illustration and interactive notebooks [15, 23], can ease the students' burdens by providing them with visualizations they can run and inspect in their browsers, and interactive programming examples that combine code and results in a standardized sandbox.

This paper will discuss the use of web technologies in scientific experiments and teaching in the context of modern web technologies and available JavaScript frameworks. The first section provided an overview over the problem domain. The second section covers current best practices in signal processing, hearing technology, and audiology, and highlights limitations of this approach. The third section illustrates how web technologies can simplify these applications, and even allow for entirely new approaches to conducting hearing experiments and teaching. The fourth section raises awareness of the technical hurdles for scientific practitioners wanting to use web technologies, and what would be needed to overcome them. Finally, the fifth chapter offers a summary and provides an outlook for a future where web technologies play an integral role in hearing experiments and science teaching.

2. HEARING EXPERIMENTS AND TEACHING

Hearing technology, and hearing-related digital signal processing are concerned with the way humans perceive sounds. As with any perceptual investigation, humans interpret sounds in many intricate and surprising ways that can be hard to model and analyze in computer systems. The following will give a short overview of the kinds of problems these areas of research are trying to solve.

For example, *Psychoacoustics* is focused on understanding the psychological and physiological responses associated with sound [17], which includes topics like sound localization, masking effects, pitch perception, and auditory scene anal-

Web Audio Conference WAC-2017, August 21–23, 2017, London, UK.

ysis. *Audiology* is concerned with hearing, balance, and related disorders in humans, and has great influence on the development and evaluation of hearing aids, cochlear implants, and assistive systems [9]. On the computational side, the broader area of *Signal Processing* includes hearing-related topics such as audio coding, noise reduction, speech recognition, musical information retrieval, or speech synthesis [20].

Since these studies are benchmarked against the capabilities of the human auditory system, they require hearing experiments for validation. These experiments typically present a test subject with acoustical stimuli, and provide a means of recording subject responses. For example, to examine auditory masking, the subject might be presented with a tone in varying levels of noise, and respond whether they could hear the tone or not. More complex setups might include assessments of the quality of synthetic speech or noise reduction, or the perception of speaker orientation in a room acoustics simulation. Typically, the stimuli will be varied according to the responses in order to arrive at some detection threshold or quality level.

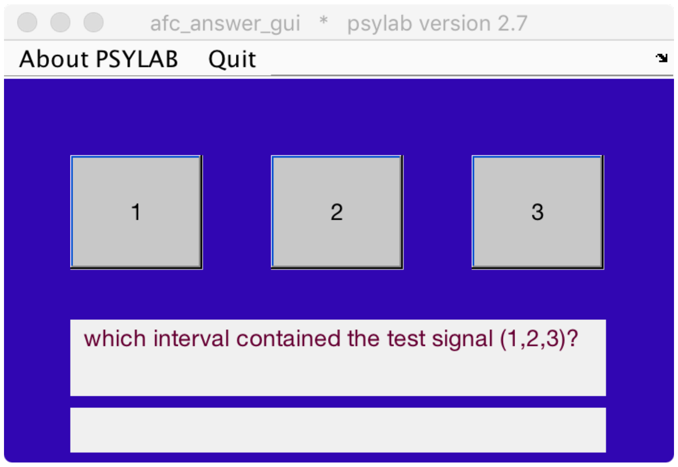
Thus, these experiments require some sort of playback capability, a data entry interface for the test subject, and some programmatic way of generating new stimuli based on past subject responses. All of this is eminently solvable with web technologies. On the hardware side, many experiments need nothing more than a pair of decent headphones in a relatively quiet environment.

Similarly, teaching audio concepts typically requires at least some playback and recording capability, and a programming environment capable of working with audio data. A big challenge in this kind of teaching is often the deployment of audio software on the students' computers. Again, web technologies can provide easy solutions to these problems.

3. CURRENT NON-WEB TECHNOLOGIES

Scientific programming in hearing related fields typically represent audio data as large, homogeneous, one- or two-dimensional arrays of floating point numbers, with convenient libraries that apply linear algebra operations on these arrays. In addition, the evaluation of experimental results requires methods for statistical data analysis and rich data visualizations. Hearing experiments and teaching examples are usually implemented as simple, utilitarian graphical user interfaces which present series of simple queries such as binary decisions or scale adjustments [5,10]. Figure 1 shows an example of a ternary forced-choice user interface from [10].

These needs are currently served by the proprietary software suites MATLAB [16] and SPSS [3], and the free and open source scientific software stacks that developed around Python [7, 19] and R [11]. All of these programming environments include strong support for fast, vectorized computations on N-dimensional arrays, and vast libraries of methods for linear algebra, statistics, digital signal processing, and data analysis. Additionally, they provide strong visualization toolboxes, GUI libraries, and notebooks. From a programming perspective, these languages are high-level interpreted, dynamically-typed languages that focus on interactive programming and rapid prototyping, and all environments include integrated development environments specifically built for scientific programming work.



which interval contained the test signal (1,2,3)?

Figure 1: Example user interface of a ternary forced-choice graphical user interface from psylab [10] using MATLAB. The user will hear three signals. For each signal, one of the three button will be highlighted in red. Only one of the three signals will contain the test signal. The user then has to click the button that contained the test signal. By changing the signal parameters over many such trials, detection thresholds can be determined.

In recent years, teaching of programming topics has been significantly helped by the introduction of *notebooks* [15,23], which are integrated documents that contain code examples and results, rich-text explanations and math, and interactive illustrations, not unlike the idea of literate programming. This concept was originally developed in 1988 with the release of Mathematica 1.0 [22], and has since spread to many interactive programming environments as *MATLAB Live Scripts* or *Jupyter Notebooks*.

While these technologies serve their purpose well for many tasks, hearing experiments and teaching in particular pose additional challenges that are not yet satisfactorily met. In both cases, a major obstacle is deployment. Both in teaching and in hearing experiments, the setup of the computers in question often requires significant technical skill and is ill-supported by the respective software suites. This often makes unsupervised experiments on subjects' own computers unfeasible. The use of web technologies would be a great boon to these kinds of scenarios, since deployment could be as easy as sending subjects or students a link to an interactive website[1].

4. CURRENT WEB TECHNOLOGIES

In order to conduct hearing experiments and teaching using web browsers, the browser needs to be able to at least

1. play and record audio signals,
2. generate and analyze audio signals,
3. record subject responses, and
4. display visualizations of scientific data.

[1]this can already be accomplished to some degree using hosted Jupyter notebooks or MATLAB Online™

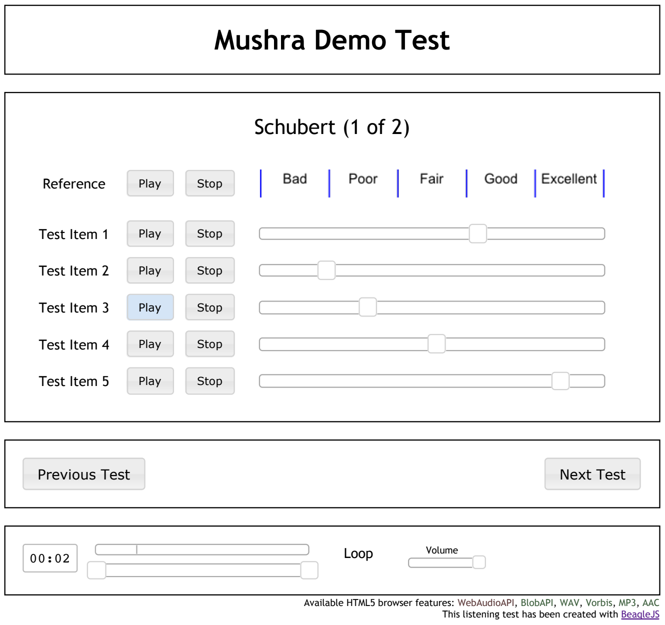


Figure 2: Example user interface of a Multi-Stimulus Test with Hidden Reference and Anchor (MUSHRA) from BeaqleJS [13]. To assess the subjective quality of audio samples, a clean reference recording is compared to several test samples, and the user can select the relative quality of each sample on a scale from *bad* to *excellent*. Unbeknownst to the user, the samples include an intentionally bad-quality *low anchor*, a medium-quality *mid anchor*, and the clean reference itself. For more details on this procedure, see [12].

In recent years, all of these requirements have been met by modern browser technologies. Regular HTML and CSS can be used to present test subjects with explanations and record subject responses. Web Audio enables real time audio playback and recording. JavaScript performance has been optimized sufficiently for real-time generation and analysis of audio signals. Software packages around SVG and Canvas were developed for creating rich, interactive data visualizations. This has been recognized by several institutions, and development has started on multiple software suites for psychoacoustic experiments in web browsers [13, 24]. An example of auch a system can be seen in Figure 2.

5. CHALLENGES

Despite the amazing technological feats that have been accomplished with web audio and modern web technologies, working with audio signals in JavaScript remains cumbersome. Most importantly, there is no built-in support for scientific data types, such as multi-dimensional arrays and complex numbers, and the facilities for applying signal processing algorithms or linear algebra to such data are very basic. While libraries exist that address these issues to some extent [2, 4, 14], they are largely incompatible with one another, and nowhere near as comprehensive as their non-web equivalents. Still, this issue could conceivably be fixed by libraries alone and does not need to be addressed at a language level.

Operator Overloading

While JavaScript can be a very pleasant programming environment, it still has a few important issues for numerical scientific code and audio applications. One such issue is JavaScript's lack of support for operator overloading. Many audio-related operations rely on the fast application of simple arithmetic expressions on homogeneous arrays. This is typically expressed in the same way as in mathematical expressions: arithmetic operations on variables containing arrays are automatically applied to all array members. Similarly, arithmetic operators are typically expected to represent matrix math if applied to variables containing matrices.

Since JavaScript currently has no operator overloading, mathematical expressions are unnecessarily cumbersome, and translation of scientific source code from other programming languages is more complex than it needs to be.

Real Time Processing

For many experiments, audio data has to be generated in real time. Even very short pauses in execution can disrupt audio streams, and introduce audible clicking artifacts. It is therefore of utmost importance that garbage collection pauses and lock contention can be controlled so as not to stall the audio processing. For serious audio applications, explicit garbage collection control, and dedicated audio threads are almost a necessity.

Audio File Download

All of the above technologies can be implemented on the client side, without requiring an interactive web server beyond a simple file server. This greatly simplifies development and deployment, and removes a large barrier to entry for many scientists and test subjects. Interactive web sites can even load audio files from the user's computer, interact with the user's microphone and loudspeaker, and generate new audio data on the fly. However, while generated audio data can be played back, it is currently not possible to allow the user to download generated audio files beyond the limits of data URIs. This would be another useful feature, particularly for teaching purposes.

Latency Control

Many audio applications require simultaneous recording and playback, for example to calculate echo effects or guitar synthesizers. It is of paramount importance that the delay between the audio hardware recording a block of audio data, and the audio hardware playing back the modified audio data is kept to an absolute minimum, as the human ear is extremely sensitive to delays in real-time audio processing. This is already being addressed in the Script Processor Nodes and Audio Workers in the current draft of the web audio APIs, although not yet widely available in release-channel web browsers. It is mentioned here only for completeness' sake.

Calibration

Finally, a fundamental issue of every non-dedicated audio environment is level calibration. Many hearing experiments require the use of calibrated levels to make sure that stimuli are never exceedingly loud, or inadequately silent. More importantly, many hearing experiments are conducted in reference to known sound pressure levels, which can be measured with a calibrated level meter, or approximated with a short hearing experiment [6].

Ultimately, however, the browser sandbox does not and can not provide enough control over the user hardware to make this perfectly robust. After all, any user can change the parameters of the audio hardware at will, without telling the browser or the operating system. There is no way around this other than always checking test results for plausibility. Still, a standardized procedure for calibrating the user's audio hardware with known minimum and maximum levels would be extremely useful for hearing experiments.

6. CONCLUSIONS

Hearing experiments and teaching can benefit greatly from incorporating web technologies and web audio into their technology stack. Several ongoing projects are actively working on advancing the state of scientific applications for web browsers, and great advances have already been reached [13, 24]. Teaching technical and scientific subjects is now almost unthinkable without modern, web-based technologies [15, 23], and active research is increasingly embracing open web technologies, as the scientific technology stacks mature.

At the same time, audio signals and digital signal processing in particular pose unique challenges to programming environments and programmers. These challenges are not yet fully solved in the world of web audio and JavaScript. In particular, native support for signal processing operations, operator overloading, latency control and audio file download would help scientific applications immensely. It is the authors' conviction that these advances will enable easier implementations of hearing experiments, better learning experiences, and ultimately, better scientific results.

7. REFERENCES

[1] Mike Bostock, Jeffrey Heer, and Vadim Ogievetsky. D3.js: Data-driven documents. http://d3js.org, 2011.

[2] Corban Brook. DSP.js, a comprehensive digital signal processing library for javascript. https://github.com/corbanbrook/dsp.js, 2010.

[3] IBM Corporation. SPSS statistics. https://www.ibm.com/us-en/marketplace/statistical-analysis-and-reporting, 1968.

[4] Jos de Jong. http://mathjs.org/, 2013.

[5] S. D. Ewert. AFC - a modular framework for running psychoacoustic experiments and computational perception models. In *Proceedings of the International Conference on Acoustics AIA-DAGA 2013*, pages 1327–1329, Merano, 2013. DEGA.

[6] Hugo Fastl, Tobias Fleischer, and Jörg Stelkens. Remote psychoacoustic experiments on audio-visual interactions. *Proc. 20th ICA Sydney*, 2010.

[7] Python Software Foundation. Python. http://www.python.org, 1991.

[8] Matthias Geier and Sascha Spors. Conducting psychoacoustic experiments with the soundscape renderer. *ITG-Fachbericht-Sprachkommunikation 2010*, 2010.

[9] S.A. Gelfand. *Essentials of Audiology.* Thieme, 2011.

[10] M. Hansen. Lehre und ausbildung in psychoakustik mit psylab: freie software für psychoakustische experimente. In *Fortschritte der Akustik – DAGA '06*, pages 591–592, Braunschweig, 2006. Dega.

[11] Ross Ihaka and Robert Gentleman. The R project for statistical computing. http://www.r-project.org, 1993.

[12] ITU-T. Method for the subjective assessment of intermediate quality level of audio systems. Recommendation BS.1534-3, International Telecommunication Union, Geneva, 10 2015.

[13] Sebastian Kraft and Udo Zölzer. BeaqleJS: HTML5 and JavaScript based framework for the subjective evaluation of audio quality. In *Linux Audio Conference*, 2014.

[14] Sébastien Loisel. Numeric javascript. http://www.numericjs.com/.

[15] McKenna R. Lovejoy and Mark A. Wickert. Using the IPython notebook as the computing platform for signals and systems courses. In *Signal Processing and Signal Processing Education Workshop (SP/SPE), 2015 IEEE*, pages 289–294. IEEE, 2015.

[16] The Mathworks. MATLAB: The language of technical computing. http://www.mathworks.com/products/matlab.html, 1984.

[17] B.C.J. Moore. *An Introduction to the Psychology of Hearing.* Emerald, 2012.

[18] University of Washington Interactive Data Lab. Datalib, a javascript data utility library. http://vega.github.io/datalib/, 2015.

[19] Travis Oliphant, Pearu Peterson, and Eric Jones et al. SciPy, a python-based ecosystem of open-source software for mathematics, science, and engineering. http://www.scipy.org, 2001.

[20] Alan V. Oppenheim and R.W. Schafer. *Digital Signal Processing.* MIT video course. Prentice-Hall, 1975.

[21] plotly. Plot.ly, *the* modern platform for agile business intelligence and data science. http://plot.ly, 2012–2017.

[22] Wolfram Research. Mathematica, the world's definitive system for modern technical computing. http://www.wolfram.com/mathematica/, 1988.

[23] Helen Shen. Interactive notebooks: Sharing the code. *Nature*, 515(7525):151, 2014.

[24] Jianchu Yao, Gregg D. Givens, and Yongbo Wan. A Web Services–Based Distributed System with Browser–Client Architecture to Promote Tele-audiology Assessment. *Telemedicine and e-Health*, 15(8):777–782, October 2009.